EXHIBIT A

Feed-forward

From Wikipedia, the free encyclopedia.
(Redirected from Feedforward)

Feed-forward is a term describing a kind of system which reacts to changes in its environment, usually to maintain some desired state of the system. A system which exhibits feed-forward behavior responds to a measured disturbance in a pre-defined way — contrast with a feedback system.

Many prerequisites are needed to implement a feed-forward control scheme: the disturbance must be measurable, the effect of the disturbance to the output of the system must be known and the time it takes for the disturbance to affect the output must be longer than the time it takes the feed-forward controller to affect the output. If these conditions are met, feed-forward can be tuned to be extremely effective.

Feed-forward control can respond more quickly to known and measurable kinds of disturbances, but cannot do much with novel disturbances. Feed-back control deals with any deviation from desired system behavior, but requires the system's measured variable (output) to react to the disturbance in order to notice the deviation.

Feed-back control is exemplified by homeostatic regulation of heartbeat in response to physical exertion. Feed-forward control can be likened to learned responses to known cues. These systems could be in control theory, physiology or computing.

Elements common to all feed-forward systems

A feed-forward system can be illustrated by comparing it with a familiar feedback system — that of cruise control in a car. When in use, the cruise control enables a car to maintain a steady road speed. When an uphill stretch of road is encountered, the car slows down below the set speed; this speed error causes the engine throttle to be opened further, bringing the car back to its original speed (a PI or PID controller would do this. Note that a good PID control *will* return the car to the original speed, after an initial transient response).

A feed-forward system on the other hand would in some way 'predict' the slowing down of the car. For example it could measure the slope of the road and, upon encountering a hill, would open up the throttle by a certain amount, anticipating the extra load. The car does not have to slow down at all for the correction to come into play.

However, other factors than the slope of the hill and the throttle setting influence the speed of the car: air temperature, pressure, fuel composition, wind speed, etc. Just setting the throttle based on a function of the slope may not result in constant speed being maintained. Since there is no comparison between the output variable, speed, and the input variable, it is not possible to resolve this problem with purely feed-forward control.

Fortunately, the two types of control are not mutually exclusive; the feed-forward system just described could be combined with the feed-back system of conventional cruise control to allow quick response with the feedback system cleaning up for any error in the predetermined adjustment made by the feed-forward system. See Model predictive control.

Feed-forward does not have the stability problems that feed-back can have. Feed-forward needs to be a pre-calibrated cause → effect, feed-back does not. There is another way of saying what was said above - that feed-forward control applies to measurable disturbances with known effects.

Physiological feed-forward system

Main article: Feed-forward (physiology)

In physiology, (also called a *feed-forward homeostatic control system*) is a homeostatic control system in which, the anticipatory effect that one intermediate exerts on another intermediate further along in the pathway allows the system to anticipate changes in a regulated variable.

Feed-forward systems in computing

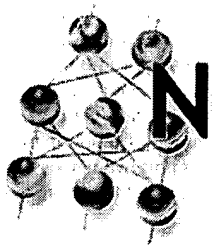
Main article: Perceptron

In computing, feed-forward normally refers to a multi-layer perceptron network in which the outputs from all neurons go to following but not preceding layers, so there are no feedback loops.

Retrieved from "http://en.wikipedia.org/wiki/Feed-forward"

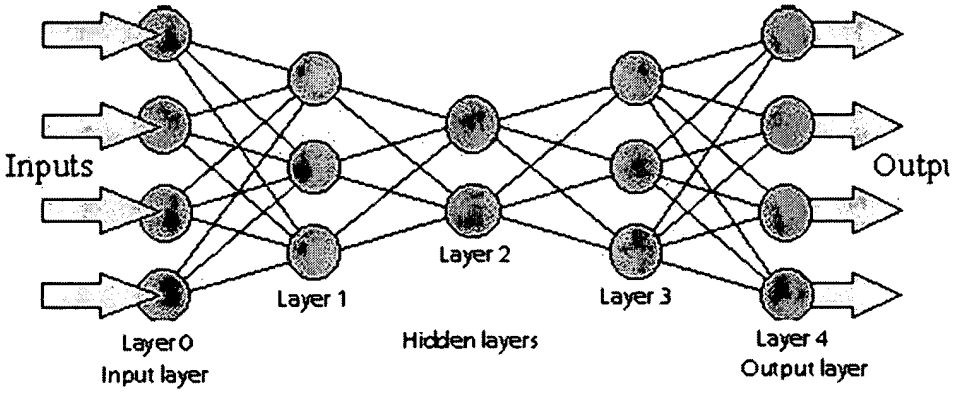
Categories: Control theory | Cybernetics | Neural networks

EXHIBIT B



Neural Networks

Feed-Forward networks:



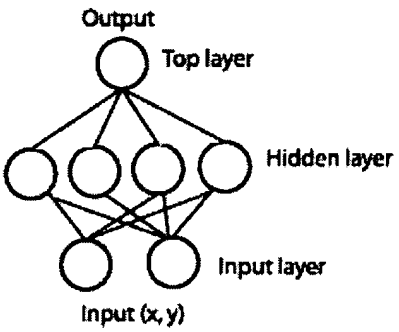
(Fig.1) A feed-forward network.

Feed-forward networks have the following characteristics:

1. Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers
2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one la to the next., and this explains why these networks are called feed-forward networks.
3. There is no connection among perceptrons in the same layer.

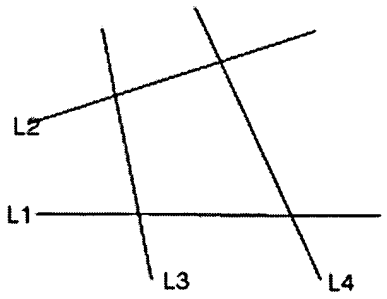
What's so cool about feed-forward networks?

Recall that a single perceptron can classify points into two regions that are linearly separable. Now let us extend the discussion into the separation of points into two regions that are not linearly separable. Consider the followi network:



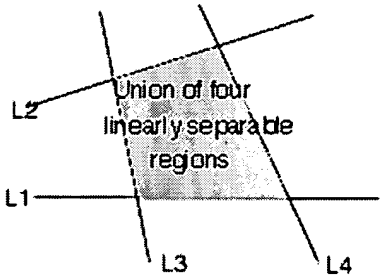
(Fig.2) A feed-forward network with one hidden layer.

The same (x, y) is fed into the network through the perceptrons in the input layer. With four perceptrons that are independent of each other in the hidde layer, the point is classified into 4 pairs of linearly separable regions, each which has a unique line separating the region.



(Fig.3) 4 lines each dividing the plane into 2 linearly separable regions.

The top perceptron performs logical operations on the outputs of the hidder layers so that the whole network classifies input points in 2 regions that mi not be linearly separable. For instance, using the AND operator on these fo outputs, one gets the intersection of the 4 regions that forms the center regi



(Fig.4) Intersection of 4 linearly separable regions forms the center region.

By varying the number of nodes in the hidden layer, the number of layers, the number of input and output nodes, one can classification of points in arbitrary dimension into an arbitrary number of groups. Hence feed-forwar networks are commonly used for classification.

Backpropagation -- learning in feed-forward networks:

Learning in feed-forward networks belongs to the realm of supervised learning, in which pairs of input and output values are fed into the network many cycles, so that the network 'learns' the relationship between the input output.

We provide the network with a number of training samples, which consists an input vector i and its desired output o. For instance, in the classification problem, suppose we have points (1, 2) and (1, 3) belonging to group 0, poi (2, 3) and (3, 4) belonging to group 1, (5, 6) and (6, 7) belonging to group 2

then for a feed-forward network with 2 input nodes and 2 output nodes, the training set would be:

{ i = (1, 2) , o =(0, 0)
i = (1, 3) , o = (0, 0)
i = (2, 3) , o = (1, 0)
i = (3, 4) , o = (1, 0)
i = (5, 6) , o = (0, 1)
i = (6, 7) , o = (0, 1) }

The basic rule for choosing the number of output nodes depends on the nun of different regions. It is advisable to use a unary notation to represent the different regions, i.e. for each output only one node can have value 1. Henc the number of output nodes = number of different regions -1.

In backpropagation learning, every time an input vector of a training sampl presented, the output vector o is compared to the desired value d.

The comparison is done by calculating the squared difference of the two:

$$\mathbf{Err = (d-o)^2}$$

The value of Err tells us how far away we are from the desired value for a particular input. The goal of backpropagation is to minimize the sum of Err all the training samples, so that the network behaves in the most "desirable' way.

Minimize $\sum \mathbf{Err = (d-o)^2}$

We can express Err in terms of the input vector (i), the weight vectors (w), the threshold function of the perceptions. Using a continuous function (inst of the step function) as the threshold function, we can express the gradient Err with respect to the w in terms of w and i.

Given the fact that decreasing the value of w in the direction of the gradien leads to the most rapid decrease in Err, we update the weight vectors every time a sample is presented using the following formula:

$$w_{new} = w_{old} - n \frac{\delta Err}{\delta w}$$

where n is the learning rate (a small number ~ 0.1)

Using this algorithm, the weight vectors are modified so that the value of E for a particular input sample decreases a little bit every time the sample is presented. When all the samples are presented in turns for many cycles, the sum of Err gradually decreases to a minimum value, which is our goal as mentioned above.

Back to Architecture